Colaboración de herramientas mediante interfaces basadas en Servicios Web: la aplicación de videoconferencia Marte

E. García[1], F. Escribano[2], C. Barcenilla[3], E. Pastor[4] y E. Barra[5]
Departamento de Ingeniería de Sistemas Telemáticos. Universidad Politécnica de Madrid
ETSI de Telecomunicación. Av. Complutense, s/n. 28040 – Madrid
Teléfono: 915 495 700 Fax: 913 367 333
E-mail: {[1]egarcia, [2]fec, [3]barcenilla, [4]encarna, [5]ebarra}@dit.upm.es

***Resumen*— El documento detalla la arquitectura ideada dentro del proyecto europeo ECOSPACE para la interoperabilidad de las herramientas de los e-Profesionales, empleando una aproximación orientada a servicios. Cada aplicación de un entorno de trabajo colaborativo debe ofrecer interfaces basadas en servicios web; en particular aquí se contempla el caso de la videoconferencia, como ejemplo representativo de sistema de funcionalidades avanzadas. Adicionalmente, los distintos servicios pueden componerse y orquestarse para ofrecer otros de mayor complejidad; para demostrar la flexibilidad y potencia de esta solución, se incluye un ejemplo que involucra múltiples herramientas. Finalmente, se contempla la posibilidad de usar otro tipo de interfaces, más extendidas actualmente, pero que implicarían un cambio profundo en la arquitectura y, por tanto, en las aplicaciones.**

***Palabras clave*— Servicios Web (*Web Services)*, Entornos de Trabajo Colaborativos (*Collaborative Working Environments*), Arquitectura Orientada a Servicios (*Service Oriented Architecture*), videoconferencia (*videoconference*)**

I. Introducción

El entorno de trabajo en las organizaciones está cambiando de forma radical, debido fundamentalmente al avance tecnológico y a la innovación en las formas de comunicación, colaboración y compartición de información. Y seguirá cambiando en los próximos años en dirección a un mundo más virtualizado, en donde la red se convertirá en el sitio de trabajo.

La posibilidad de colaboración ubicua, independiente de tiempo y espacio, entre equipos de trabajo de una organización o entre organizaciones, aportará la flexibilidad necesaria para poder reaccionar a ese entorno en constante cambio y será esencial para hacer el mejor uso del conocimiento y competencias disponibles[1]. Una colaboración eficiente tiene un fuerte impacto en la creatividad y productividad.

Así, los entornos para el soporte a la colaboración y para el trabajo colaborativo (*Collaborative Working Environments*, CWE) están emergiendo en muchos ámbitos de actividades de trabajo cotidianas y la actual oferta de aplicaciones y herramientas es muy amplia y variada.

Uno de los problemas más importantes de la infraestructura de colaboración con el que se enfrentan los usuarios es la falta de interoperabilidad entre las distintas aplicaciones. En general las aplicaciones están diseñadas pensando que el grupo que colabora utilizará la misma aplicación. Sin embargo los miembros de un grupo están típicamente involucrados en múltiples actividades y proyectos en paralelo, colaborando con distintos equipos y utilizando en cada caso diferentes herramientas y aplicaciones.

Debido a la falta de interoperabilidad[2], es casi imposible para un usuario tener integrada toda la funcionalidad y disponer de un único entorno conceptual de trabajo colaborativo. Como consecuencia de ello, para iniciar una videoconferencia y discutir un documento en grupo, por ejemplo, el usuario tendrá que configurar, arrancar y manejar una a una varias aplicaciones diferentes. Si las aplicaciones estuvieran integradas de alguna manera, bastaría simplemente con un clic para desencadenar todo ese proceso.

El presente artículo describe una solución al problema de la interoperabilidad entre un grupo heterogéneo de herramientas, basándose en la creación de interfaces estandarizadas y abiertas, en el marco del proyecto europeo ECOSPACE[3]. El trabajo queda dividido como sigue: como primer paso, la sección II concreta una arquitectura a la que deberán adaptarse todas las aplicaciones objeto de este trabajo, efectuando una categorización de los servicios que serán ofrecidos, y que permitirán, en base a su composición flexible, ofrecer distintas capacidades. Dentro de estos servicios, en la sección III, se recoge el caso particular de Marte[4], aplicación avanzada de videoconferencia, sobre la que se plasmarán las ideas propuestas. Para demostrar la viabilidad del enfoque, en la sección IV, se presenta un ejemplo de creación de servicios complejos mediante orquestación de los más básicos, incluyendo distintas herramientas. Para finalizar, la sección V plantea la orientación a recursos como una solución alternativa, si bien teniendo en cuenta que implica cambios profundos a todos los niveles.

II. Integración de herramientas colaborativas

El punto de partida de ECOSPACE es la problemática citada, y un grupo de herramientas heterogéneo, que constituyen el porfolio habitual de los usuarios de los entornos de trabajo colaborativos. Por ello, en las secciones siguientes se identifica una arquitectura común, y posteriormente se diseñan las interfaces adecuados para lograr la interoperabilidad.

A. *Arquitectura para entornos de trabajo colaborativo*

Para resolver el problema de la interoperabilidad de las distintas aplicaciones dentro de un mismo entorno de trabajo colaborativo, se ha optado por diseñar en ECOSPACE una Arquitectura de Referencia basándose en el modelo de la orientación a servicios (SOA)[5]. Esta arquitectura representa un modelo en el que cada proceso (computacional) es descompuesto en distintos subprocesos: el conjunto de subprocesos compone el proceso completo, pero cada uno de ellos puede ser distribuido y reutilizado, dada la independencia que deben mantener entre ellos. La granularidad o complejidad de los subprocesos (en adelante, Servicios Básicos de Colaboración, o simplemente servicios) no está prefijada; el criterio es que se ofrezcan con interfaces bien definidas, indicando al menos nombre del servicio y los tipos de datos de entrada y salida. Una arquitectura así está basada, por tanto, en tres elementos: servicios, su descripción, y los mensajes que se intercambian. Sus principios básicos son:

- Bajo nivel de acoplamiento entre servicios, garantizando su independencia de la lógica y el funcionamiento de otros servicios; tampoco deben guardar estado.
- Abstracción: los servicios ocultan sus procesos internos al mundo exterior.
- Reusabilidad: la idea principal tras la descomposición es fomentar la reutilización.
- Composición: servicios básicos pueden componerse y formar servicios más complejos (como se verá en la sección IV).
- Descubrimiento: para que los servicios sean usados de manera extendida, no sólo deben estar bien descritos a nivel de funcionalidad, sino que deben proveerse mecanismos para su localización, generalmente mediante registros.

La arquitectura concreta dentro de ECOSPACE se muestra en la Fig. 1, donde se detallan los niveles que la componen: cuatro horizontales y tres verticales.

Las capas horizontales son:

- Capa de Servicios Básicos: los servicios básicos (BCS) componen los bloques más pequeños, atómicos, que serán usados en el resto de capas. Se definen como aquellas tareas colaborativas que no pueden ser divididas en unidades más simples. Los servicios que existen en esta capa son ofrecidos por las tecnologías básicas de trabajo colaborativo (algunas de las cuales se verán en la sección II-B, como el correo electrónico, la videoconferencia, o los repositorios compartidos), si bien las herramientas comunes (aplicaciones de uso diario) pueden incluir muchas de estas tecnologías. La manera más extendida de ofrecer estos servicios básicos es mediante Servicios Web, aunque un primer paso es ofrecerlos mediante XML-RPC.

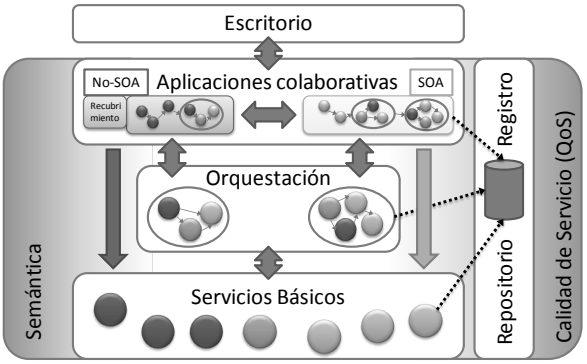

Fig. 1. Arquitectura de ECOSPACE para Entornos de Trabajo Colaborativo

- Capa de Orquestación: los servicios orquestados quedan definidos como conjuntos de servicios básicos que son ejecutados en un orden establecido para proporcionar al usuario una funcionalidad colaborativa de valor añadido. Dentro de ECOSPACE, son conocidos como Servicios Compuestos de Colaboración (*Composite Collaborative Services* – CoCoS).
- Capa de Aplicaciones Colaborativas: la componen las aplicaciones software que usan las capacidades de los servicios básicos y/o los orquestados, según los requisitos del usuario. Al mismo tiempo, pueden ser las mismas que ofrezcan servicios básicos al resto.
- Capa de Escritorio: la interfaz de usuario destinada a los e-Profesionales que usarán las herramientas de colaboración.

Al mismo tiempo, se identifican una serie de capas verticales, que dan valor añadido a todas las demás:

- Capa de Infraestructura Semántica: almacena modelos y perfiles de usuarios, metadatos, información de contexto y reglas que usarán el resto de capas para proporcionar inteligencia y personalización. Incluye el marcado semántico de CoCoS, mediante ontologías de Web Semántica, o SIOC[6]
- Capa de Registro/Repositorio: almacena la información de todos los componentes (CoCoS, BCS y aplicaciones) en una base de datos –directorio– donde pueda ser encontrada para su uso y composición.
- Capa de Calidad de Servicio (QoS): provee capacidades de monitorización, seguridad, gestión de errores, transacciones, escalabilidad y confiabilidad.

La arquitectura abarca, de manera genérica, la mayoría de las aplicaciones actuales: incluye aquellas aplicaciones ya existentes, la cuales serán descompuestas y expuestas según los criterios del nivel inferior, y permite crear nuevas aplicaciones que cubran las expectativas de los profesionales de las aplicaciones colaborativas.

B. *Definición de interfaces para aplicaciones colaborativas*

Anteriormente se han definido los Servicios Básicos (BSC) como los componentes más pequeños de un Entorno de

Trabajo Colaborativo. La diversidad de servicios existente es producto de su origen: la mayoría son ofrecidos desde un número elevado de aplicaciones complejas ya existentes. Por ejemplo, en el caso de Marte –aplicación de multiconferencia– se ofrecen principalmente servicios de vídeo-llamada, pero también de presencia, o de mensajería instantánea. Al mismo tiempo, una mensajería similar es ofrecida por otra aplicación como Skype. Alcanzar la interoperabilidad deseada entre servicios dada esta heterogeneidad de funcionalidades hace necesario que las interfaces estén muy definidas, siguiendo estándares en lo posible. Dentro de ECOSPACE, se ha optado por usar Servicios Web, junto con todas las tecnologías asociadas a ellos: SOAP[7], WSDL[8] y UDDI[9], que especifican, respectivamente, el formato, la descripción y el registro y búsqueda.

Dentro del proyecto ECOSPACE, y para el ámbito de los entornos de trabajo colaborativo, se han identificado numerosos servicios básicos que son ofrecidos mediante alguna tecnología –entendida como la abstracción de más alto nivel que caracteriza una aplicación– , y mediante un análisis preliminar se han agrupado según características comunes. La lista realizada sólo puede servir como base, ya que el número de servicios se incrementa día a día, pero contribuye a orientar el esfuerzo hacia las capacidades más necesarias. Algunos de los servicios identificados se recogen en la Tabla I a modo de ejemplo.

TABLA I
TECNOLOGÍAS, SERVICIOS BÁSICOS Y APLICACIONES IDENTIFICADAS EN ECOSPACE

Tecnología	**Servicios básicos**	**Aplicaciones**
MultimediaConference	createConference, changeConfMode, inviteParticipant, …	Marte, Skype…
SharedWorkspaces	addDocument, getDocument, search…	BCSW, BC, SAP Netweaver…
InstantMessaging	sendMessage, receiveMessage…	Post-@, Skype…
Email	createMail, sendMail…	BSCW, Outlook, Thunderbird…
PresenceAndAvailability	join, leave, getStatus…	Post-@, Jabber…
UserManagement	createNewUser, getUserData…	Active Directory, BSCW, Lotus Notes…

Los servicios básicos presentados están asociados a tecnologías específicas, que pueden ser interpretadas como componentes o funcionalidades de aplicaciones software. De hecho, la relación es "muchos-a-muchos" entre las tecnologías y las aplicaciones. En la arquitectura, esto se traduce en que la capa de Servicios Básicos es poblada por componentes provenientes de la capa de Aplicaciones Colaborativas. Para las aplicaciones que ya cuentan con una Arquitectura Orientada a Servicios es inmediato hacer esta asociación, simplemente exponiendo sus funciones a través de interfaces. Sin embargo, dado que este paradigma es relativamente reciente, muchos servicios están aún "bloqueados" en el interior de aplicaciones que sólo permiten acceder a ellos mediante sus propias interfaces de usuario no estandarizadas. Por ello, se proponen recubrimientos, que permitirán una adaptación de esas aplicaciones no SOA a la arquitectura propuesta. En el caso de Marte (como en muchas otras herramientas), ya se ofrecía funcionalidad orientada a servicios, pero no mediante Servicios Web, de modo que se diseñó un recubrimiento especial, como se verá en las secciones III-C y III-D.

Al mismo tiempo, la arquitectura está pensada para que las aplicaciones de la capa superior usen los servicios básicos, provengan éstos de aplicaciones SOA o no-SOA. En el caso de aplicaciones orientadas a servicios, no se necesita ningún requisito especial: podrán enriquecer sus capacidades directamente con la composición de servicios. Pero aplicaciones no orientadas a servicios necesitarán "adaptadores" para interoperar, lo cual queda fuera del ámbito de este trabajo; en las siguientes secciones se tratará sólo el primer tipo de aplicaciones, SOA.

III. SERVICIO DE VIDEOCONFERENCIA; MARTE

Para la demostración práctica de los principios de colaboración mencionados anteriormente se ha elegido la aplicación Marte, desarrollada dentro de nuestro grupo de trabajo. Dicha herramienta, que ofrece servicios de conferencia multimedia, ha sido adaptada a la arquitectura descrita, implementando un conjunto de Servicios Web como se describe a continuación.

A. Arquitectura de Marte

Marte es un sistema de videoconferencia y compartición de aplicaciones basado en el protocolo SIP. Las conferencias soportadas por Marte permiten a los participantes cambiar el modo en el que se le presentan los medios que recibe del servidor; si algún participante cambia el modo de interacción, los demás participantes reciben ese mismo cambio, de forma que el esquema mostrado para todos ellos es el mismo. El servidor es flexible en cuanto a estas configuraciones, permitiendo mostrar el vídeo de un único participante, el vídeo de varios simultáneamente (otorgando prioridad a uno si así se desea), y compartir el escritorio de uno o varios participantes.

La arquitectura del servidor Marte puede verse en la Fig. 2:

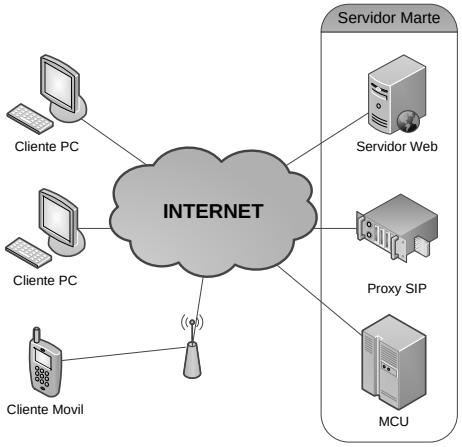

Fig. 2. Arquitectura de la aplicación Marte

Los elementos de Marte son:

- Un **servidor web** que permite el registro de usuarios, así como la descarga de las aplicaciones cliente.
- Un **registrador SIP** (*proxy*), basado en SER[10] que se encarga de autenticar a los usuarios, encaminar las llamadas y proporciona el servicio de presencia.
- Una **unidad de control multipunto** (*MCU*) que se encarga de hacer la suma de audio, el mosaico de vídeo y la transcodificación necesaria para soportar distintos clientes.

Existen actualmente dos versiones de la aplicación: una para escritorio, y otra basada en web, ofreciendo ambas la misma implementación de los Servicios Web.

B. Parlay X

Para definir los servicios que debe ofrecer la aplicación de videoconferencia, se ha tomado como referencia el trabajo del consorcio Parlay[11]. Parlay ha definido junto con ETSI[12] y el 3GPP[13] una API para permitir a los desarrolladores de aplicaciones acceder a los servicios de una red pública de telecomunicaciones. Una de las principales fortalezas de la API de Parlay es la independencia de red y tecnológica, puesto que históricamente la provisión de este tipo de servicios ha estado sujeta a una tecnología de red específica. Parlay X es un subconjunto de la tecnología Parlay que permite el acceso a los servicios de la red a través de una interfaz basada en Servicios Web, necesaria para el caso de ECOSPACE. Los servicios ofrecidos por la especificación Parlay X pretenden ser de alto nivel y sencilla utilización. La versión actual de la especificación fue publicada en marzo de 2005. Actualmente se encuentran disponibles borradores de la versión 3.0.

En el caso de conferencias multimedia Parlay define la especificación "*OSA Parlay X Web Services, Part 12: Multimedia Conference*"[14] que es la que se decidió implementar, aunque enriquecida para permitir el acceso a las funcionalidades más avanzadas del servidor Marte. Los servicios implementados permiten la creación y destrucción de conferencias, la gestión de participantes y modos de interacción de la conferencia, información sobre presencia de usuarios y servicios específicamente desarrollados para integración con otras herramientas.

C. Desarrollo de interfaces XML-RPC

Como un primer paso hacia la exposición de interfaces basándose en una orientación a servicios, y dado que otras herramientas dentro del consorcio ya hacían uso de XML-RPC, se desarrolló una API con esta tecnología, pero teniendo en cuenta que debería dar exactamente los mismos servicios que los que se pretendía dar a través de Servicios Web. Puesto que el servidor está desarrollado en Java se utilizó la implementación de Apache[15] de XML-RPC que permite incorporar de manera muy simple un servidor en cualquier aplicación.

El modelo de servicio está basado en las siguientes entidades:

- **Conferencia**: contexto identificado de forma unívoca en el que los participantes intercambian flujos multimedia y pueden ser añadidos o desconectados.
- **Participante**: cada una de las partes implicadas en una conferencia. Un caso especial de participante es el creador *(owner)* de la conferencia que es quien puede terminar la conferencia.
- **Medios**: cada uno de los flujos de datos intercambiados por los participantes en una conferencia (audio, vídeo, escritorio, etc.).

Los servicios creados se resumen a continuación, y están descritos en su totalidad en el Apéndice:

- Gestión de conferencias (permiten arrancar y terminar conferencias): *CreateConference*, *EndConference*, *GetConferenceInfo.*
- Gestión de participantes (empleados para invitar y desconectar participantes a una conferencia en curso): *AddParticipant, GetParticipants, GetParticipantInfo, DisconnectParticipant.*
- Gestión avanzada de conferencia (se encargan de la gestión de funcionalidades avanzadas de las videoconferencias de Marte): *ChangeConferenceMode*, *AddMediaForParticipant,RemoveMediaForParticipant*
- Presencia (informan sobre la presencia de usuarios en el sistema): *GetUserList, GetPresenceStatus.*

D. Recubrimiento de WS

Tal y como se ha explicado, en la primera iteración, se implementó la API descrita como servicios XML-RPC; sin embargo, la aproximación de Servicios Web estandarizados (basados en SOAP/WSDL) es la escogida para este proyecto, ya que permite el uso de registros y la composición y orquestación de servicios de forma dinámica.

Con el fin de mantener ambas APIs operativas y ofrecer por ambas la misma funcionalidad básica se ha desarrollado un recubrimiento (*wrapper*) que atiende las peticiones SOAP y las traduce a peticiones XML-RPC tal y como se ve en la Fig. 3.

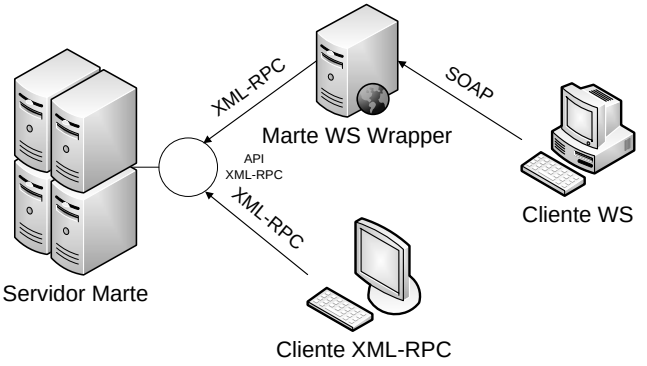

Fig. 3. Recubrimiento de Servicios Web para Marte

Con esta solución es posible atender a peticiones a través de ambas interfaces, y permite la migración gradual de los clientes de un servicio a otro. Otra ventaja viene del hecho de que las herramientas de desarrollo de Servicios Web facilitan mucho la creación de servicios para ser desplegados sobre contenedores de aplicaciones J2EE, como Tomcat[16]; implementarlos

directamente en el servidor Marte sería mucho más complejo.

Los servicios ofrecidos son totalmente equivalentes y para facilitar su utilización se ha respetado en la medida de lo posible la nomenclatura de los métodos así como los parámetros y valores de retorno. La independencia de los distintos componentes mostrados en la Fig. 2 ha permitido de hecho el cambio de la arquitectura del servidor Marte sin afectar a la implementación de los Servicios Web, ni al resto de clientes de los servicios de videoconferencia de Marte.

IV. Composición de servicios

A. Mecanismos de orquestación

La capa de orquestación está situada justo por encima de la de servicios básicos. En ella, los BCS se combinan en Servicios Compuestos de Colaboración, o CoCoS (ya definidos en la sección II-A). Dentro de ECOSPACE, todos los CoCoS pasan a formar parte de un repositorio para que puedan ser usados por las aplicaciones, de manera independiente, combinados entre ellos, o incluyendo otros servicios básicos. Puede apreciarse la flexibilidad de la aproximación, que permite conseguir prácticamente cualquier funcionalidad a partir de los servicios disponibles. Un Servicio Compuesto sólo incluye dos tipos de información:

- El comportamiento esperado, como cualquier otro servicio básico
- El flujo de trabajo y la lógica asociada a la ejecución de los servicios básicos

El proceso de crear un CoCoS es denominado orquestación de servicios, y puede realizarse en tiempo de diseño (orquestación estática) o de ejecución (orquestación dinámica). La aproximación aquí mostrada, y generalizada en ECOSPACE, es híbrida: se diseñan plantillas abstractas, que describen el flujo de trabajo general para la funcionalidad deseada, pero no especifican qué servicios (básicos o compuestos) concretos deben utilizarse. En tiempo de ejecución, los mecanismos de descubrimiento, usando las capas verticales de la arquitectura (Registro y Semántica), serán responsables de localizar e incluir en la plantilla los servicios necesarios.

Dado que los servicios básicos están definidos mediante interfaces de Servicios Web, la descripción de los CoCoS puede realizarse mediante WS-BPEL[17], una tecnología madura para definir orquestación de servicios. Los flujos BPEL generalmente incluyen puntos de control para ramificaciones, opciones para proceso en paralelo, interacción con humanos… y, adicionalmente, permiten exponer interfaces de Servicios Web para que, a su vez, puedan ser usados como servicios compuestos.

B. Caso de uso: composición usando Marte

Para demostrar la utilidad y flexibilidad de este enfoque, se ha diseñado un CoCoS que hace uso de los servicios básicos definidos para la aplicación Marte, además de los servicios ofrecidos por otra aplicación (BSCW[18]), y algunos servicios compuestos ya existentes. El objetivo de este Servicio Colaborativo Compuesto, denominado Subir y Discutir Documento en Videoconferencia (*Upload and Discuss Document in Videoconference*), es proporcionar un medio para que un grupo de usuarios discutan, a través de videoconferencia, un documento subido a un repositorio. Para ello, tras subir el documento objeto de la discusión, se manda una notificación a los usuarios indicando dónde está el documento, y la conferencia se inicia con aquellos que están conectados. Al mismo tiempo, el documento se abre, y la aplicación de videoconferencia se activa en modo de escritorio compartido, para que todos puedan interaccionar con él. Este CoCoS aprovecha dos CoCoS previamente definidos dentro del proyecto: Subir Documento y Notificar (*UploadDocumentAndNotify*, que sube un documento a un repositorio y manda una notificación al respecto) y Crear Videoconferencia (*CreateVideoconference*, que crea una videoconferencia con un grupo dado de usuarios). Adicionalmente, emplea directamente servicios básicos de la tecnología de Conferencia Multimedia (*MultimediaConference*). La relación del CoCoS con los distintos servicios, dentro de la arquitectura de referencia para entornos de trabajo colaborativos se muestra en la Fig. 4.

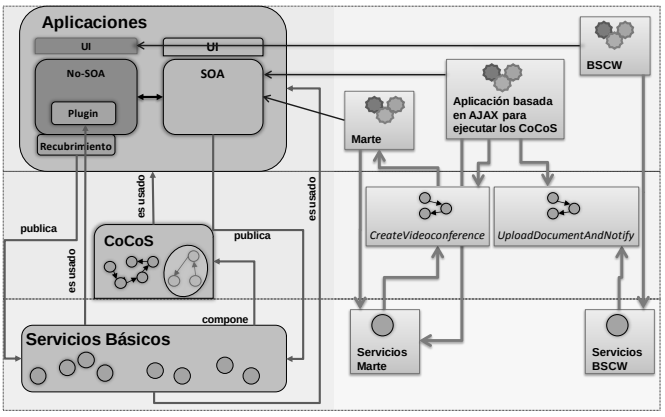

Fig. 4. Relación del CoCoS con la arquitectura de referencia de ECOSPACE

El CoCoS está modelado como un único proceso, y su proceso de ejecución es como sigue:

1. Se solicita la lista de usuarios registrados a la aplicación de videoconferencia (en nuestro caso, Marte), con el Servicio Web *GetUserList*.
2. El usuario escoge el documento a subir, y la carpeta del BSCW donde debe almacenarse.
3. El usuario indica la lista de gente interesada en el documento: aquellos que recibirán la notificación, y participarán (caso de estar conectados a Marte) en la discusión por videoconferencia.
4. Se invocan los CoCoS ya definidos de *CreateVideoconference* y *UploadDocumentAndNotify*, con sus parámetros correspondientes.
5. El documento se abre en el escritorio del usuario que ejecuta el CoCoS y se indica a la aplicación de

videoconferencia que debe establecer el modo en escritorio compartido, usando el Servicio Web *ChangeConferenceMode*.

El uso de BPEL en este CoCoS permitiría ofrecerlo a su vez como otro CoCoS (más complejo en cuanto a funcionalidad) a través de las interfaces normalizadas de Servicios Web.

V. Evolución de las interfaces: REST

Una posible vía de evolución de estas interfaces para lograr interoperabilidad con más plataformas podría estar ligada a transformarlas en servicios REST orientados a recursos. Las interfaces de servicios web XML-RPC y SOAP/W3C (también conocidos como *Big Web Services*) de los CoCoS, y de Marte en particular, están basadas en el modelo de llamadas a procedimientos remotos (*RPC*), donde se definen una serie de primitivas para que otras aplicaciones interaccionen con sus servicios.

La interfaz actual es parte de un modelo para la interacción de aplicaciones de colaboración bien definido y completamente basado en RPC (la Arquitectura para Entornos de Trabajo Colaborativos descrita en la sección II-A), y como tal es capaz de integrarse plenamente en un flujo de trabajo basado en esta arquitectura.

En oposición al modelo RPC en el que están basados los servicios web SOAP/W3C, la arquitectura orientada a recursos (*ROA*)[19], basada en REST[20], tiene como eje central al recurso: cualquier cosa que sea lo suficientemente importante para ser referenciada en sí misma a través de un Identificador de Recurso Uniforme (*URI*), que constituirá su nombre. Los recursos poseen las características generales de la Web, teniendo por tanto que ser referenciables, no mantener estado, poseer una representación y, en la misma, hacer referencia a otros recursos.

Según sus proponentes, ROA consiste en organizar recursos para componer servicios poderosos, pero conceptualmente simples y accesibles desde un gran número de clientes estándar. Sostienen que con el concepto de servicio web se tiende a pensar en SOAP, WSDL y la pila WS-* (*Big Web Services*), pero que este método, a pesar de su nombre, en la práctica no funciona como la Web. Por el contrario, la Web se basa en URIs que apuntan a recursos y en enlaces entre ellos, pero los *Big Web Services* sólo exponen una URI y no hacen uso de enlaces. Además, utilizan pobremente las características de HTTP. Los *Big Web Services* no se benefician de la orientación a recursos del modelo web: no son direccionables, *cacheables*, bien conectados, no hacen uso de una interfaz uniforme, y suelen mantener estado. En la práctica, además, tienden a tener problemas de interoperabilidad con distintos clientes.

Por el contrario, en ROA los recursos deben exponer una interfaz uniforme acorde con el modelo web, lo que implica que se puedan realizar operaciones sobre los mismos sólo a través de los métodos uniformes definidos para la web, como GET, PUT, POST, DELETE, HEAD y OPTIONS. De acuerdo con el modelo web, los recursos deben tener además dos propiedades: seguridad (*safety*) e idempotencia. La seguridad indica que una operación GET o HEAD no cambia al recurso ni modifica su estado, mientras que la idempotencia que la repetición de una misma operación sobre el recurso siempre deja a éste en el mismo estado (por ejemplo, si el recurso se elimina a través de DELETE dos veces, en ambos casos el resultado es el mismo: el objeto desaparece). Las características de seguridad e idempotencia son importantes porque el medio por el cual se cursan las operaciones, la web, no es confiable y ante fallas del mismo las operaciones pueden repetirse sin consecuencias inesperadas.

Los *Big Web Services* están pensados para implementar servicios orientados a proceso a través de intermediarios (*brokers*), aplicaciones más usuales en ambientes de negocios y gobierno, y menos en áreas académicas y técnicas, y en la web.

Una posible implementación de una interfaz ROA en Marte para implementar servicios colaborativos compuestos demandaría la definición de un modelo de recursos que reemplace al conjunto de primitivas que actualmente cuenta la interfaz de tipo RPC expuesta a través de XML-RPC y SOA/W3C. La principal ventaja de esta interfaz sería la posibilidad de acceso a Marte desde clientes livianos (como los que pueden ejecutarse en un navegador web usando JavaScript), haciendo posible de esta forma la construcción de *widgets*/*gadgets* que hagan uso de los recursos expuestos por Marte de forma eficiente.

Afortunadamente la interfaz de servicios exportada por Marte (ver Apéndice), a pesar de estar realizada en el modelo RPC, ya tiene características de orientación a recursos (por ejemplo *conference*, *participant*, *user*, *media*) por lo que su reformulación en base al modelo ROA consistiría en centrar el modelo en estos recursos y asignar las operaciones a los distintos métodos HTTP.

La característica de direccionamiento de los recursos permitiría implementar servicios externos de directorios de usuarios o sesiones a través de una interfaz uniforme. Sin embargo, no todas las características de REST serían de utilidad, por ejemplo, la posibilidad de cachear los recursos no sería aprovechada significativamente por Marte dado que los recursos que se expondrían a través de la interfaz cambiarían a menudo e individualmente no serían consultados por un número de clientes que suponga la aparición de problemas de escalabilidad.

VI. Conclusiones

El trabajo realizado presenta una posible solución a la comunicación entre aplicaciones de trabajo colaborativo, y muestra la aptitud del enfoque mediante uno de los diversos casos prácticos que se están implementando en el proyecto ECOSPACE. Sin embargo, se detectan también algunas debilidades; aparte de las constricciones inherentes al modelo de orientación a servicios, sería necesario considerar los aspectos de seguridad y de gestión de usuarios. La composición de servicios básicos para dar lugar a herramientas

de colaboración enriquecidas debe estar controlada en tanto en cuanto puede desearse que no todas las aplicaciones sean accesibles desde cualquier otra, o sólo si cumplen con ciertos requisitos de seguridad; debe incidirse en que los datos que se intercambian los servicios podrían ser sensibles, e involucrar a distintas organizaciones. A este respecto, la gestión de identidad dentro de todo el sistema es crítica para poder proporcionar a los usuarios un punto de acceso único, y que el uso de los distintos servicios (ofertados generalmente por distintas aplicaciones) sea transparente a las capas superiores de la arquitectura. El trabajo de ECOSPACE va ahora en estas líneas, para conseguir que la colaboración propuesta sea real fuera de los meros entornos de investigación.

APÉNDICE: SERVICIOS BÁSICOS DE MARTE

A continuación se detalla la interfaz de acceso a los servicios básicos de videoconferencia proporcionados por la aplicación Marte. Se incluye el identificador de cada servicio, una breve descripción de su funcionalidad, y los parámetros de entrada y salida, así como su tipado.

CreateConference	
Descripción	Crea una conferencia.
Parámetros de entrada	**ConferenceType**: *array* de *strings* que especifican los medios que se utilizarán en la conferencia (audio, video, chat, vnc…). **ConferenceDescription**: *string* que describe la conferencia. **MaximunDuration**: duración máxima de la conferencia en minutos. **MaximunParticipants**: número máximo de participantes. **ConferenceOwner**: *string* que identifica al creador de la conferencia que tiene permitido cambiar la configuración.
Valor de retorno	*String* que identifica de forma unívoca la conferencia recién creada.

EndConference	
Descripción	Destruye una conferencia.
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor.
Valor de retorno	*Boolean* que indica si la conferencia ha sido cerrada con éxito.

GetConferenceInfo	
Descripción	Recupera información sobre una conferencia en curso.
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor.
Valor de retorno	Objeto que almacena los parámetros de creación de la conferencia.

ChangeConferenceMode	
Descripción	Cambia el modo de interacción de una conferencia.
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor. **ConferenceMode**: *string* que especifica el nuevo modo de interacción. **Participants**: *array* de *strings* que contiene una lista de identificadores de usuario para configurar el modo especificado.
Valor de retorno	*Array* de *strings* que contiene el modo aplicado y una lista de participantes que componen el modo especificado.

GetParticipants	
Descripción	Obtiene la lista de participantes en una conferencia.
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor.
Valor de retorno	*Array* de *strings* que contiene los identificadores de los participantes.

InviteParticipant	
Descripción	Invita un participante a una conferencia
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor. **Participant**: *string* que identifica al participante.
Valor de retorno	*Boolean* que indica si la invitación ha sido enviada con éxito.

DisconnectParticipant	
Descripción	Desconecta a un participante de una conferencia.
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor. **Participant**: *string* que identifica al participante.
Valor de retorno	*Boolean* que indica si el participante ha sido desconectado con éxito.

AddMediaForParticipant	
Descripción	Añade un flujo multimedia a un participante en una conferencia.
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor. **Participant**: *string* que identifica al participante. **Media**: *string* que identifica el nuevo medio. **Direction**: *string* que indica la dirección del nuevo flujo (*in*, *out*, etc).
Valor de retorno	Ninguno.

RemoveMediaForParticipant	
Descripción	Elimina un flujo multimedia a un participante en una conferencia
Parámetros de entrada	**ConferenceId**: *string* que identifica la conferencia en el servidor. **Participant**: *string* que identifica al participante. **Media**: *string* que identifica el medio **Direction**: *string* que indica la dirección del flujo (*in*, *out*, etc).
Valor de retorno	Ninguno.

GetUserList	
Descripción	Obtiene la lista de usuarios registrados en el servidor.
Parámetros de entrada	Ninguno.
Valor de retorno	*Array* de *strings* que contiene los identificadores de los usuarios.

GetPresenceStatus	
Descripción	Dice si un usuario está conectado o no al sistema.
Parámetros de entrada	**Participant**: *string* que identifica al participante.
Valor de retorno	*Boolean* que indica si el usuario está conectado.

AGRADECIMIENTOS

Este trabajo ha sido realizado dentro del Proyecto Integrado ECOSPACE (eProfessional Collaboration Space), parcialmente financiado por la Comisión Europea dentro del 6º Programa Marco: FP6-IST-5-35208. Los autores quieren agradecer a los miembros del proyecto por su exitoso desarrollo.

Asimismo, los autores quieren agradecer a Carlos Bueno y a Miguel Gómez las valiosas ideas y contribuciones que han tenido en el campo de realización de este artículo.

REFERENCIAS

[1] J. Mohr y R. Spekman. "Characteristics of Partnership Success: Partnership Attributes, Communication Behavior, and Conflict Resolution Techniques". Strategic Management Journal, Vol. 15 (2), 135-152, 1994

[2] M. Koza y A. Lewin. "Managing Partnerships and Strategic Alliances: Raising the Odds of Success". European Management Journal, 18(2), 146-151, 2000

[3] eProfessional Collaborative Workspace (ECOSPACE), http://www.ip-ecospace.org

[4] J. Cerviño, P. Rodríguez, J. Salvachúa, G. Huecas y F. Escribano, "Marte 3.0: Una videoconferencia 2.0", Actas de las VII Jornadas de Ingeniería Telemática (JITEL), septiembre 2008

[5] S. Hashimi, "Service-Oriented Architecture Explained", O'Reilly ONDotnet.com, agosto 2003

[6] John G. Breslin, Andreas Harth, Uldis Bojars, Stefan Decker: Towards Semantically-Interlinked Online Communities. In Asunción Gómez-Pérez, Jérôme Euzenat (Eds.): The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings. Lecture Notes in Computer Science 3532 Springer 2005, ISBN 3-540-26124-9: 500-514

[7] "Simple Object Access Protocol 1.1", http://www.w3.org/TR/SOAP

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, "Web Services Definition Language", http://www. w3.org/TR/wsdl, marzo 2001

[9] T. Bellwood, L. Cl´ement, D. Ehnebuske, A. Hately, Maryann Hondo, Y.L. Husband, K. Januszewski, S. Lee, B. McKee, J. Munter, and C. von Riegen. UDDI Version 3.0, julio 2002

[10] SIP Express Router, http://www.iptel.org/ser

[11] H. Lofthouse, M.J. Yates and R. Stretch, "Parlay X Web Services", BT Technology Journal, Volume 22, Number 1, p. 81-86, 2004

[12] European Telecommunications Standards Institute (ETSI), http://www.etsi.org

[13] 3rd Generation Partnership Project (3GPP), http:// www.3gpp.org

[14] Open Service Access (OSA) Parlay X Web Services – Part 12: Multimedia conference (Release 7), 3GPP TS 29.199-12 v7.1.1, diciembre 2007

[15] Apache XML-RPC, http://ws.apache.org/xmlrpc

[16] Apache Tomcat, http://tomcat.apache.org

[17] Business Process Execution Language for Web Services (BPEL), Version 1.1. http://www.ibm.com/developerworks/library/ws-bpel, mayo 2003

[18] R Bentley, T Horstmann, K Sikkel, J Trevor, "Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System", Proceedings of the 4th International WWW Conference, 1995

[19] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures", tesis doctoral, University of California, Irvine, 2000

[20] L. Richardson, Sam Ruby, "RESTful Web Services", O'Reilly, 2007, ISBN 0-596-52926-0